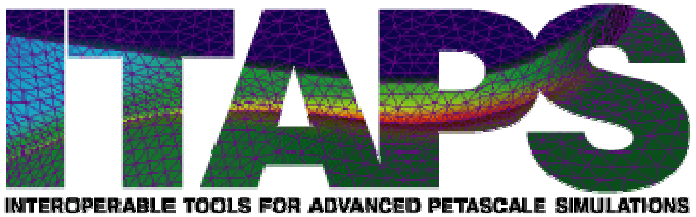
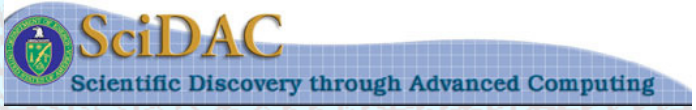


Interoperable Tools for Advanced Petascale Simulations (ITAPS)

ComPASS Meeting Update

December 3, 2008

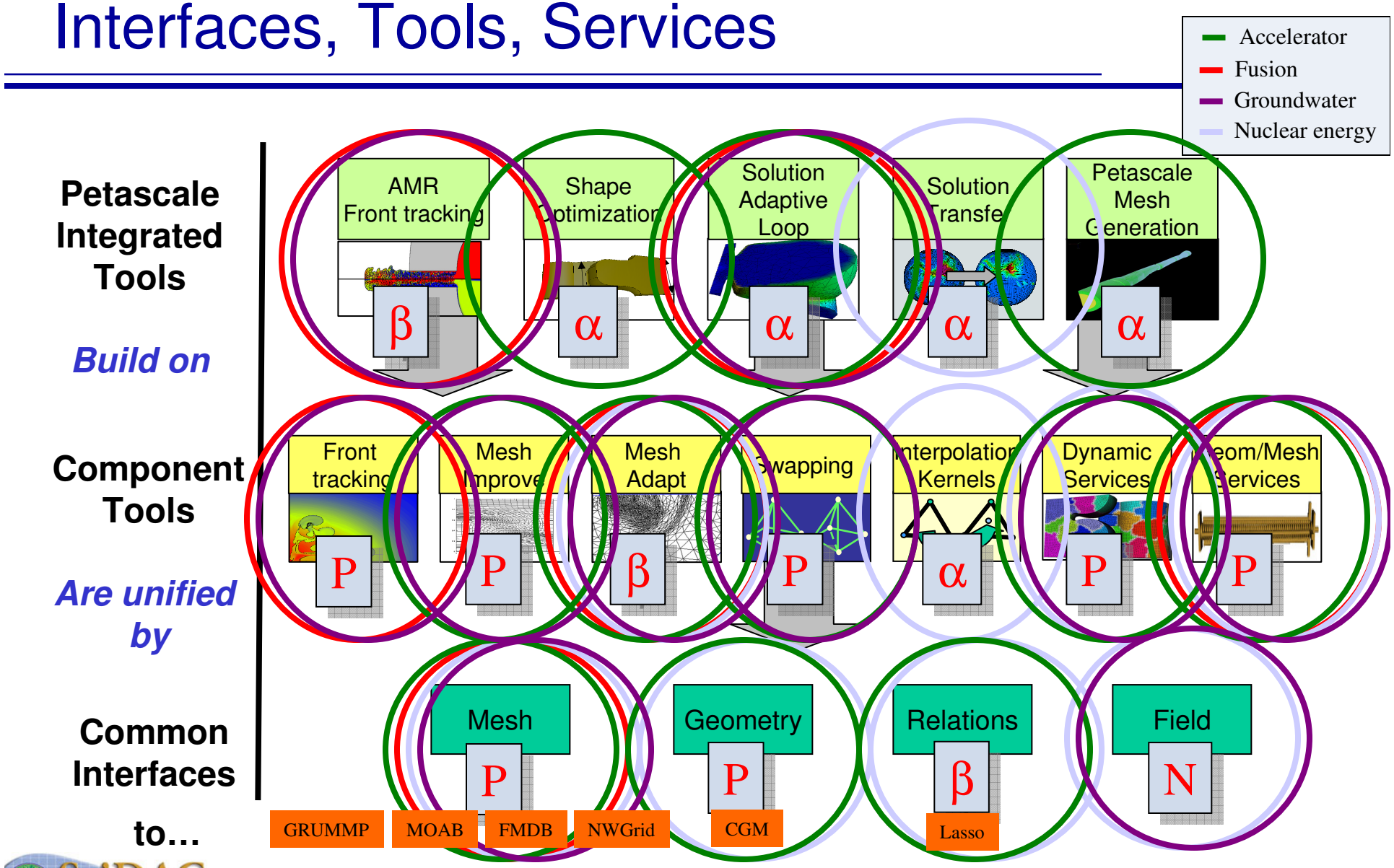



LLNL-PRES-407129

Outline

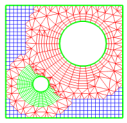
- ITAPS data models & interfaces
- ITAPS services
- ANL/ITAPS activities & opportunities

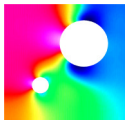
ITAPS 3-Tier Approach Interfaces, Tools, Services



SciDAC
Scientific Discovery through Advanced Computing

The ITAPS data model abstracts PDE-simulation data hierarchy





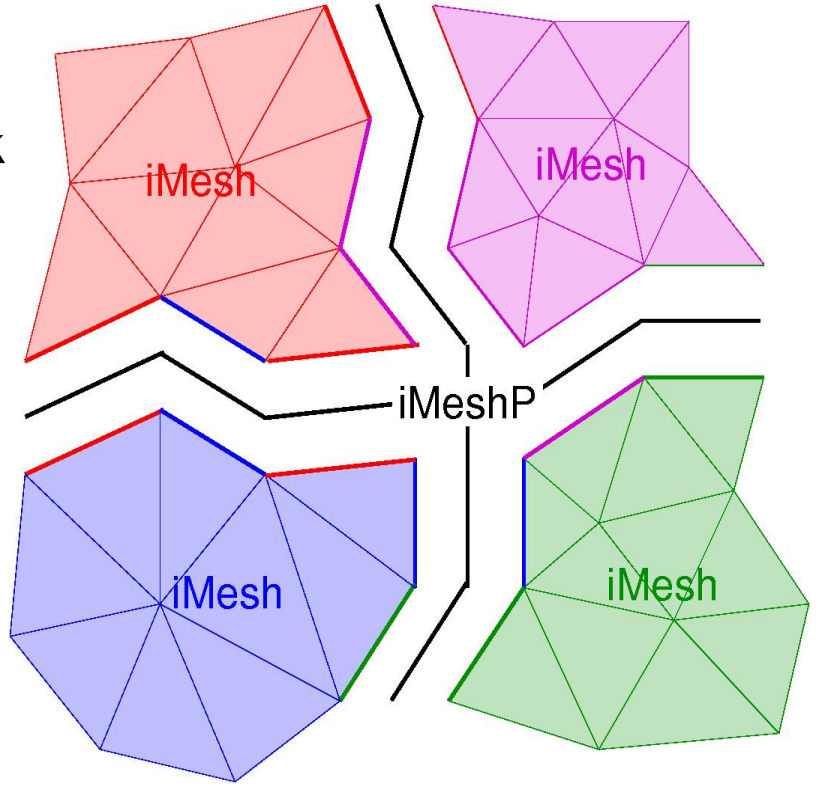
- Core Data Types
  - *Geometric Data*: provides a high level description of the boundaries of the computational domain; e.g., CAD, image, or mesh data (iGeom)
  - *Mesh Data*: provides the geometric and topological information associated with the discrete representation of the domain (iMesh)
  - *Field Data*: provides access to time dependent physics variables associated with application solution. These can be scalars, vectors, tensors, and associated with any mesh entity. (iField)
- Data Relation Managers (iRel)
  - Provides control of the relationships among the core data types
  - Resolves cross references between entities in different groups
  - Provides functionality that depends on multiple core data types

The ITAPS data model has four fundamental "types"

- *Entity:* fine-grained entities in interface (e.g., vertex, face, region)
- *Entity Set:* arbitrary collection of entities & other sets
  - Parent/child relations, for embedded graphs between sets
- *Interface Instance:* object on which interface functions are called and through which other data are obtained
- *Tag:* named datum annotated to Entities and Entity Sets

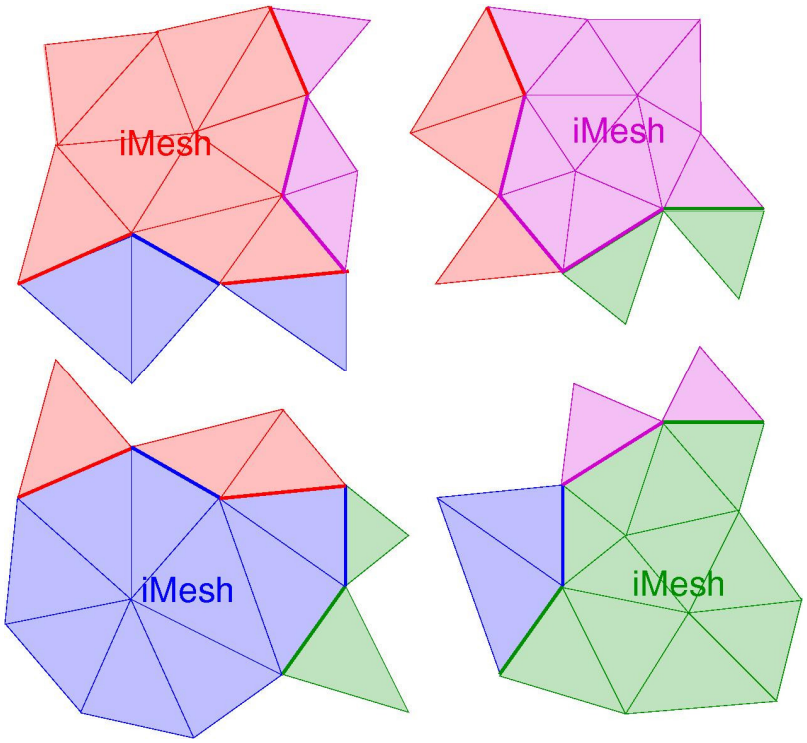
The iMeshP parallel interface defines a partition model

- *Process*: a program executing; MPI process
  - # of processes == MPI_Comm_size
  - Process number == MPI_Comm_rank
- *iMesh instance*: mesh database provided by an implementation
  - One or more instances per process
- *Partition*: describes a parallel mesh
  - Maps entities to subsets called *parts*
  - Maps parts to processes
  - Has a communicator associated with it



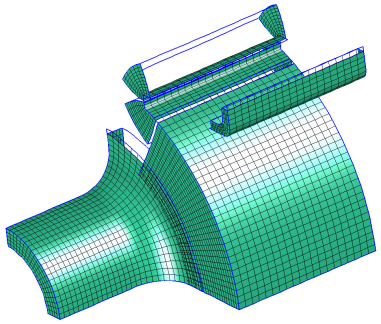
The Partition Model

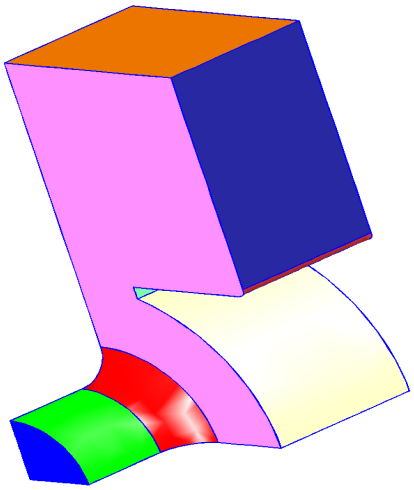
- *Ownership*: right to modify an entity
- *Internal entity*: Owned entity not on an interpart boundary.
  - E.g., all triangles w/ same color as iMesh label for part
- *Part-Boundary entity*: Entity on an interpart boundary
  - E.g., bold edges
  - *Shared* between parts (owner indicated by color; other parts have copies).
- *Ghost entity*: Non-owned, non-part-boundary entity in a part
  - E.g., triangles whose color is different from iMesh label
  - Needed for adjacency and/or solution data
- *Copies*: ghost entities + non-owned part-boundary entities.



Basic and advanced functionalities are supported in the geometry interface

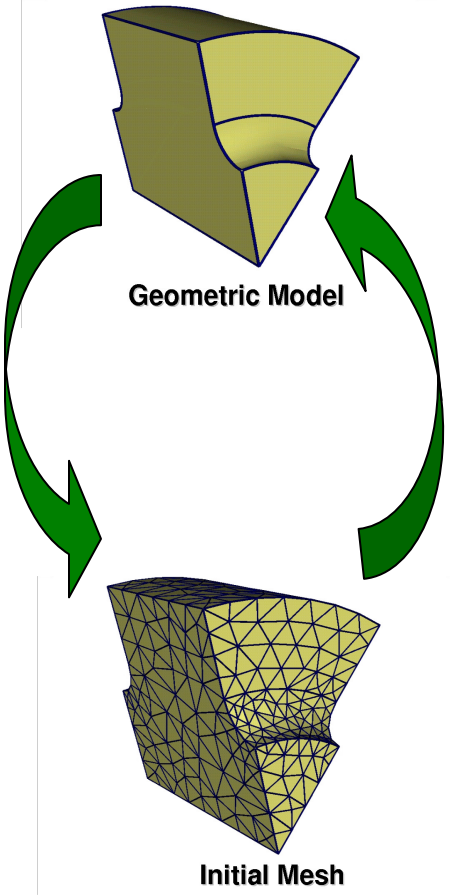
- Model loading and initiation
- Topological queries of entities and adjacencies
- Pointwise geometric shape interrogation
- Parametric coordinate systems
- Model topology modification
- Model construction (primitives, booleans, transforms, etc.)



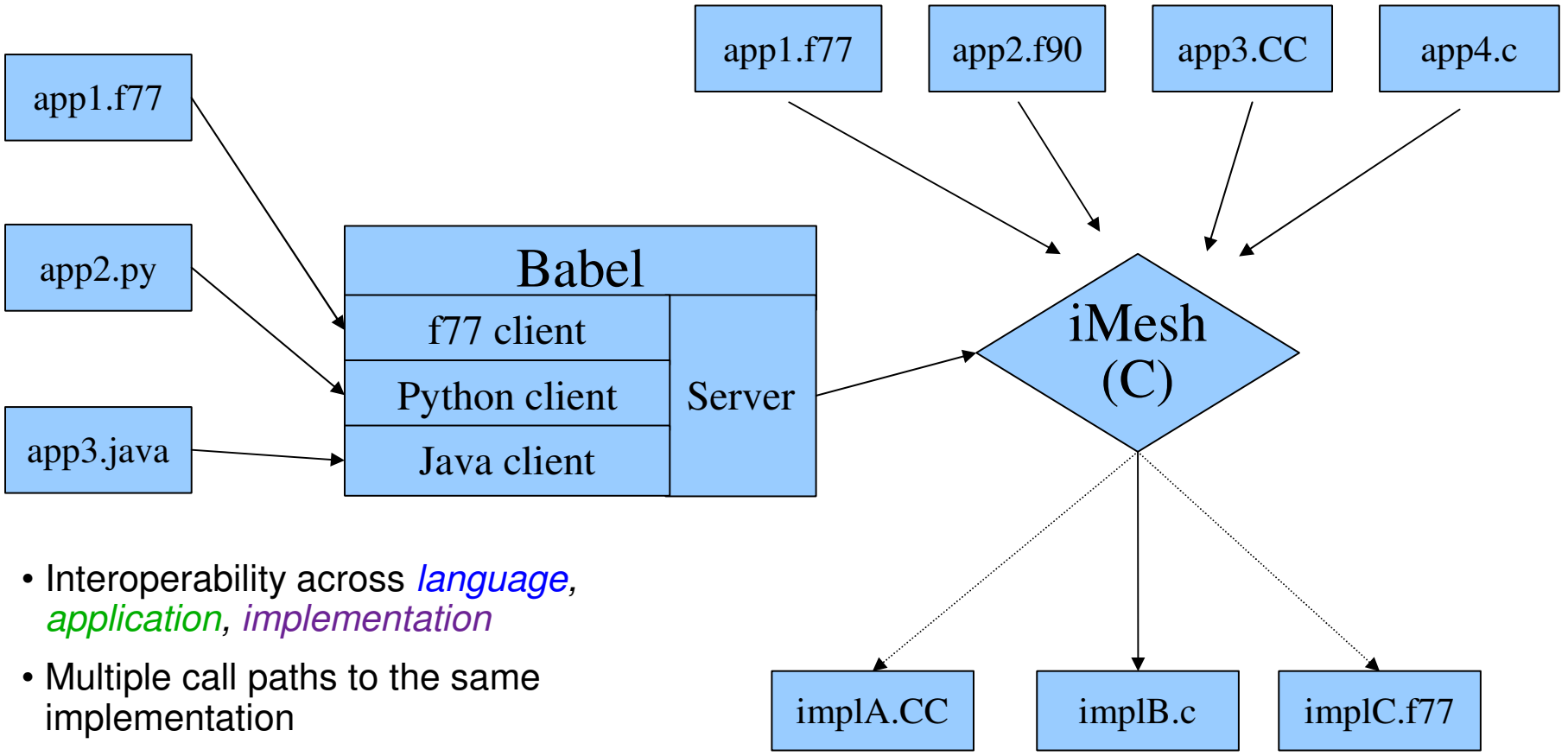


Relating mesh and geometry data is critical for advanced ITAPS services

- Required for e.g., adaptive loops, mesh quality improvement
- Mesh/Geometry Classification Interface
  - Manages the relationship between the high level geometric description and the mesh
  - Called by an application that knows about both
- Capabilities
  - For a given mesh entity, get the geometric entity against which it is classified
  - Establish a classification relationship between a mesh entity and a geometric entity



ITAPS Interfaces Designed for Interoperability



- Interoperability across *language*, *application*, *implementation*
- Multiple call paths to the same implementation
- Efficiency preserved using direct, C-based interface

SciDAC
Scientific Discovery through Advanced Computing

Simple Example: HELLO iMeshP

Revised 11/08

```
#include "iMesh.h"
#include "iMeshP.h"
#include <mpi.h>
int main(int argc, char *argv[]) {
  char *options = NULL;
  iMesh_Instance mesh;
  iMeshP_PartitionHandle partition;
  int me, dim, num, ierr, options_len=0;
  iBase_EntitySetHandle root;
  /* create the Mesh instance */
  iMesh_newMesh(options, &mesh, &ierr, options_len);
  iMesh_getRootSet(mesh, &root, &ierr);

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &me);
1 /* Create the partition. */
  iMeshP_createPartitionAll(mesh, MPI_COMM_WORLD, &partition, &ierr);

  /* load the mesh */
2 iMeshP_loadAll(mesh, partition, root, argv[1], options, &ierr,
          strlen(argv[1]), options_len);

  /* Report number of Parts in Partition */
  iMeshP_getNumParts(mesh, partition, &num, &ierr);
3 printf("%d Number of Parts = %d\n", me, num);
...
```

Parallel Version: HELLO iMeshP

1) *Instantiates Partition*
2) *Reads mesh into mesh instance and Partition*
3) *Reports # parts in Partition*

HELLO iMeshP Makefile

Revised 11/08

```
CC = mpicc -g

IMESH_DIR = /usr/local/itaps/FMDB_iMeshP
include $(IMESH_DIR)/lib/iMesh-Defs.inc

INCPATH += $(IMESH_INCLUDES)
LIBS += $(IMESH_LIBS)

hello_iMeshP: hello_iMeshP.c
	$(CC) $(INCPATH) -c hello_iMeshP.c
	$(CC) -o $@ hello_iMeshP.o $(LIBS)
```

ITAPS API's: Argument Handling Conventions

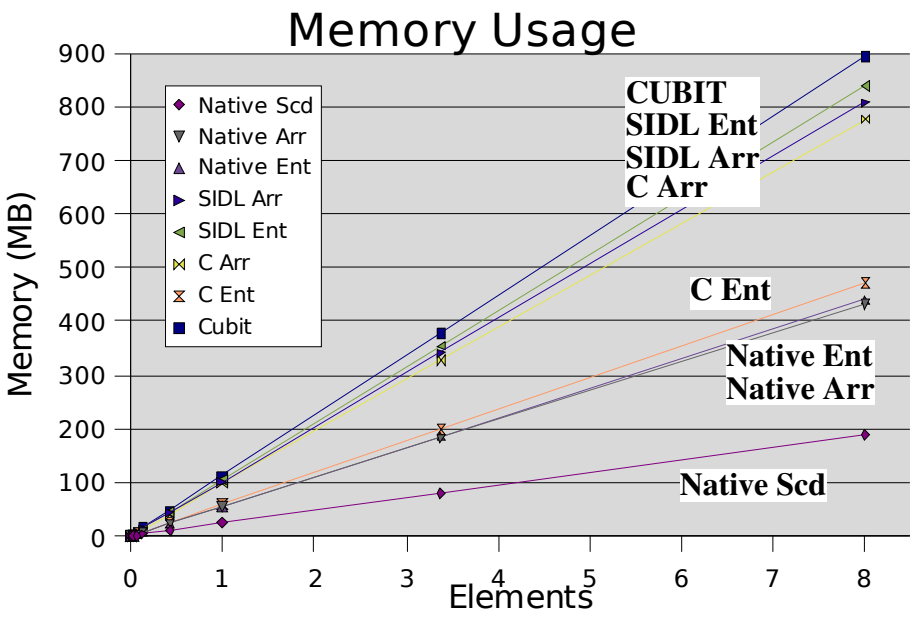
- ITAPS API's are C-like and can be called directly from C, Fortran, C++
- Arguments pass by value (in) or reference (inout, out)
  - Fortran: use %VAL extension
- Memory allocation for lists done in application *or* implementation
  - If inout list comes in allocated, length must be long enough to store results of call
  - By definition, allocation/deallocation done using C malloc/free; application required to free memory returned by implementation
  - Fortran: Use “cray pointer” extension (equivalences to normal f77 array)
- Handle types typedef'd to size_t (iBase_EntityHandle, iBase_EntitySetHandle, iBase_TagHandle, iMesh_Instance)
- Strings: char*, with length passed by value after all other args
- Enum's: *values (iBase_SUCCESS, etc.) available for comparison operations, but passed as integer arguments*
  - *Fortran: named parameters*

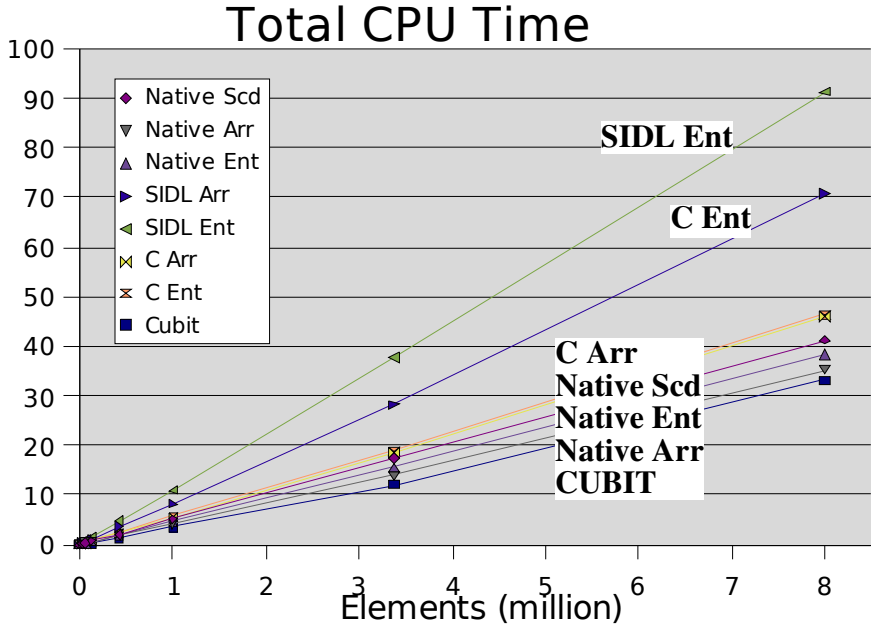
Argument Handling Conventions

Issue	C	FORTRAN	SIDL
Function Names	iXxxx_ prefix	Same as C	Removed iXxxx_ prefix; SIDL interface organization
Interface Handle	Typedef'd to size_t, as type iXxxx_Instance; instance handle is 1st argument to all functions	#define'd as type Integer; handle instance is 1st argument to all functions	Interface type derived from sidl.BaseInterface
Enumerated Variables	All arguments integer-type instead of enum-type; values from enumerated types	Same, with enum values defined as FORTRAN parameters	Int-type arguments; enumerated types defined in iXxxx:: namespace, and values appear as iXxxx::enumName_enumValue
Entity, Set, Tag Handles	Typedef'd as size_t; typedef types iBase_EntityHandle, iBase_EntitySetHandle, iBase_TagHandle	#define'd as type Integer	Handles declared as SIDL opaque type (mapped to void* in C/C++ server)
Lists	• In: X *list, int occupied_size • Inout: X **list, int *allocated_size, int **occupied_size • malloc/free-based memory allocation/deallocation	Same, with Cray pointers used to reference arrays (see FindConnectF example	• In: sidl::array<X> list, int occupied_size • Inout: sidl::array<X> &list, int &occupied_size • sidl::array class memory allocation
String	char*-type, with string length(s) at end of argument list	char[]-type without extra length argument (this length gets added implicitly by FORTRAN compiler)	sidl::string type without extra length argument

Performance

- Large applications balance memory and cpu time performance
- Implementations of iMesh vary on speed vs. memory performance
  - Create, v-E, E-v query, square all-hex mesh
  - Entity- vs. Array-based access
- Compare iMesh (C, SIDL), Native (MOAB), Native Scd (MOAB), CUBIT
  - Ent-, Arr-based access
  - All-hexahedral square structured mesh





Performance in building a finite element stiffness matrix

- Set up a simple stiffness matrix for a 2D diffusion equation
- Examine costs of entity access via native data structures, arrays, entity iterators and workset iterators
- Arrays minimize time overhead but require a data copy
- Entity iterators are straightforward to program, minimize memory overhead, but maximize time cost
- Entity array iterators balance time/memory tradeoffs but are the most difficult to program

$$\nabla^2 u = f$$

$$u(x=0)=1 \; u(x=1) = 1$$

$$u_y(x=0, x=1) = 0$$

	Time (ms)	iMesh Overhead
Native	10479	
Array-based	10774	2.8%
Entity Iterator	11642	11.1%
Workset Iterator (1)	11351	8.3%
Workset Iterator (3)	11183	6.7%
Workset Iterator (5)	11119	6.1%
Workset Iterator (10)	11095	5.8%
Workset Iterator (20)	11094	5.8%

Performance of iMesh Swap for 3D Meshes

- Comparing GRUMMP native implementation to service with GRUMMP iMesh implementation
- Most remaining overhead is in transcribing data to return format expected by iMesh

Case	# of Tets	Native		iMesh	
		Swaps	Rate (1/s)	Swaps	Rate (1/s)
Rand1	5104	10632	29500	10838	21300
Rand2	25704	65886	27700	67483	22100
Airplane	251140	25448	3380	28629	2800
Rocket	464080	53331	3540	59330	2790

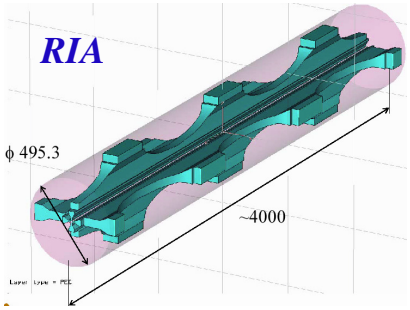
Performance of Zoltan Partitioning for 3D Meshes

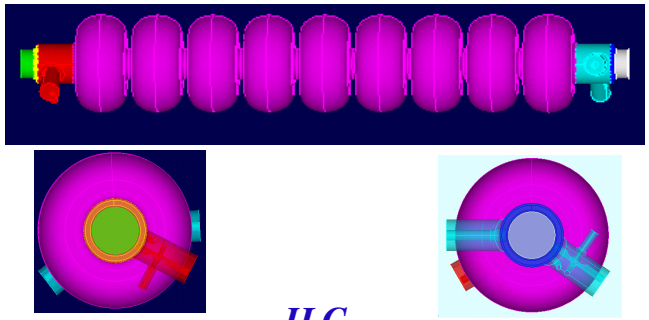
- Comparing MOAB native implementation linking to Zoltan partitioning service with the MOAB iMesh implementation
- Using a coordinate bisection geometric partitioner on tetrahedral meshes and array-based access to the data

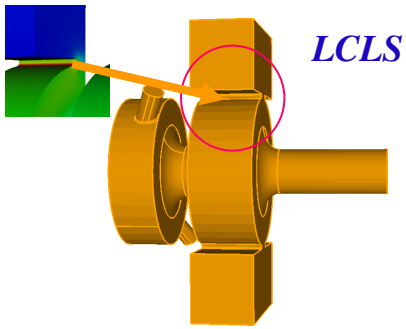
Number of Tets	Native (sec)	iMesh (sec)	iMesh Overhead
15591	0.866	0.869	0.35%
20347	0.971	0.976	0.51%
34750	1.28	1.31	2.34%
54383	1.72	1.75	1.74%
100630	2.76	2.82	2.17%

Shape Optimization for Accelerator Cavity Design

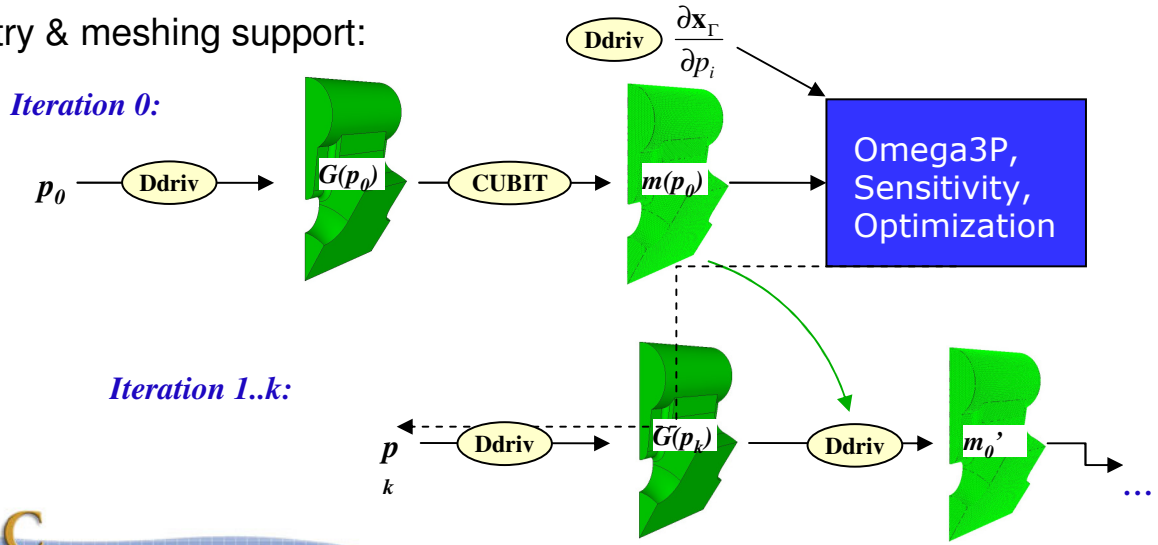
- Optimizing a cavity design is still mostly a manual process
- Future accelerators employ complex cavity shapes that require optimization to improve performance







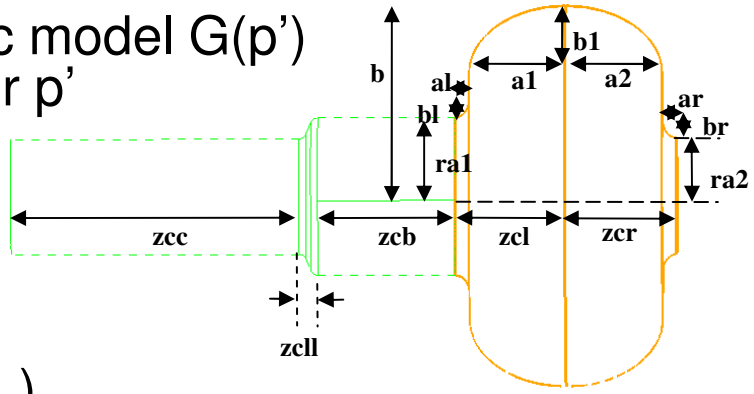
- Geometry & meshing support:

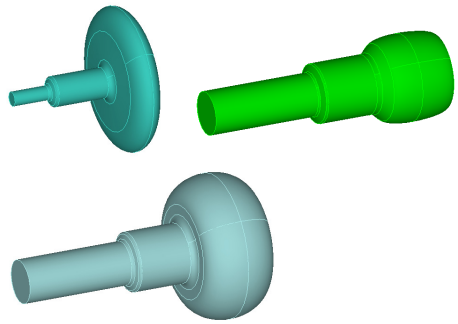


Fixed mesh topology:
Convergence
No re-meshing
Re-use factorization

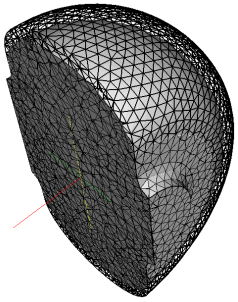
Shape Optimization for Accelerator Cavity Design

- Generate new geometric model G(p') given a parameter vector p'
  - *MkILCCell function*
  - DDRIV
  - CGM (iGeom)

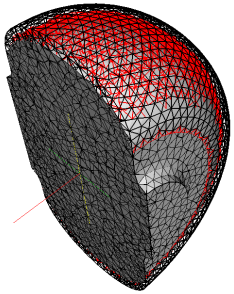




- Associate old mesh $m(p_o)$ to new geometry G(p'), project to CAD
  - DDRIV
  - CGM (iGeom)
  - MOAB (iMesh)
  - LASSO (iRel)

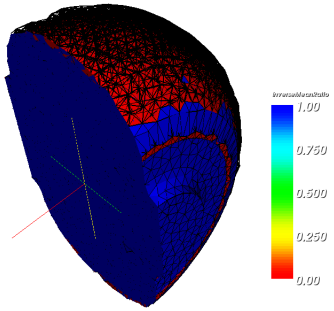


New geom, old mesh

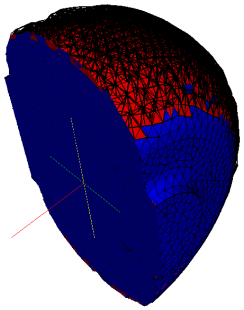


Project to CAD, inverted elements

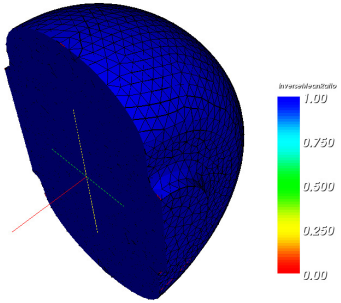
- Smooth mesh
  - …
  - Mesquite



Smooth Curves



Smooth Surfaces



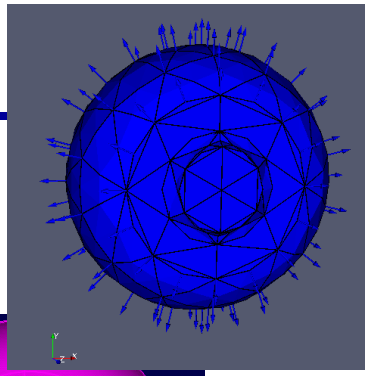
Smooth Volume

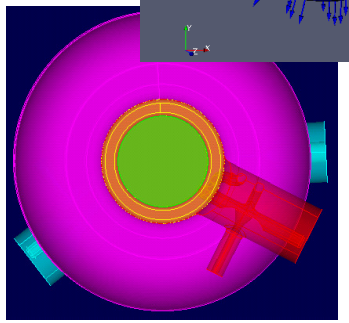
SciDAC
Scientific Discovery through Advanced Computing

Services Provided by DDRIV

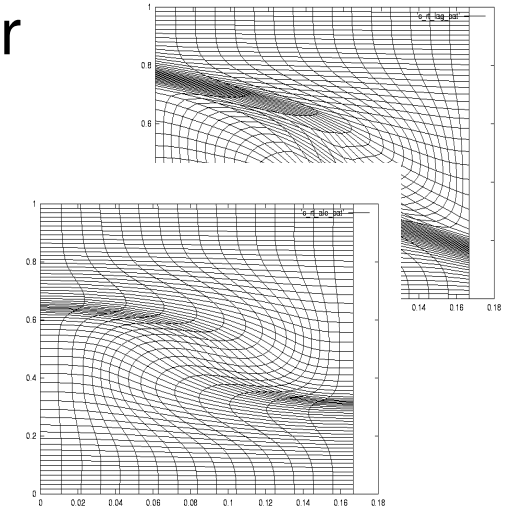
- Parameterized geometric model construction
  - You write function which constructs model using iGeom
  - DDRIV acts as driver and handles IO
- Coordination of mesh smoothing on geometric model
- Re-classification of “old” mesh on “new” model
- Target matrix-based smoothing of re-classified mesh
- Computation of design velocities & embedding on mesh using iMesh Tags

Mesquite provides advanced mesh smoothing capabilities

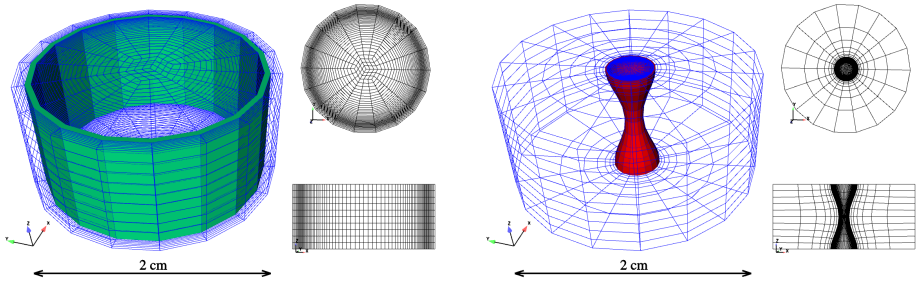




- Mesquite is a comprehensive, stand-alone library for mesh quality improvement with the following capabilities
  - Shape Quality Improvement
  - Mesh Untangling
  - Alignment with Scalar or Vector Fields
  - R-type adaptivity to solution features or error estimates
    - Maintain quality of deforming meshes
    - Anisotropic smoothing
    - Control skew on mesh boundaries
- Uses node point repositioning schemes



Our mesh quality improvement work has impacted many DOE applications



Application: Plasma implosion using ALE methods

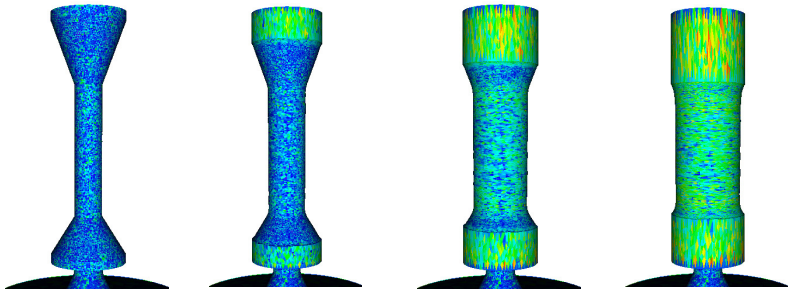
Challenge: Maintain good mesh quality and biasing during deformation of plasma.

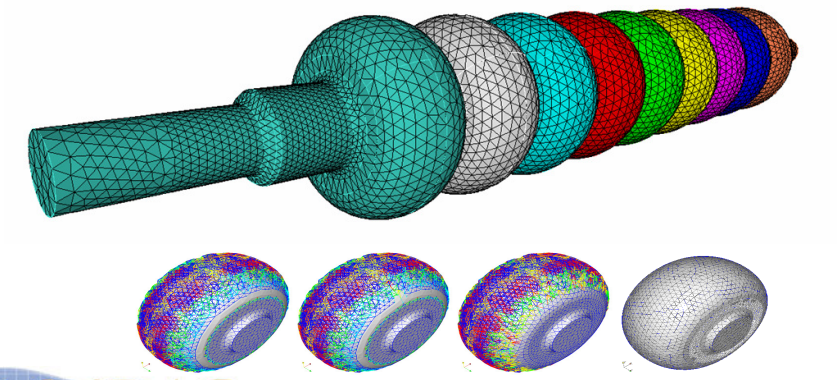
Impact: Prior to use of Mesquite, this calculation could not be performed by Alegra due to ineffective mesh rezoning algorithm.

Application: Burn of rocket propellants in a time-deforming domain

Challenge: Maintain good tetrahedral element shape quality as domain deforms

Impact: Condition number smoother (through ShapeImprovementWrapper) enabled many burn simulations at CSAR/UIUC.





Application: Shape optimization for accelerator cavities to minimize losses

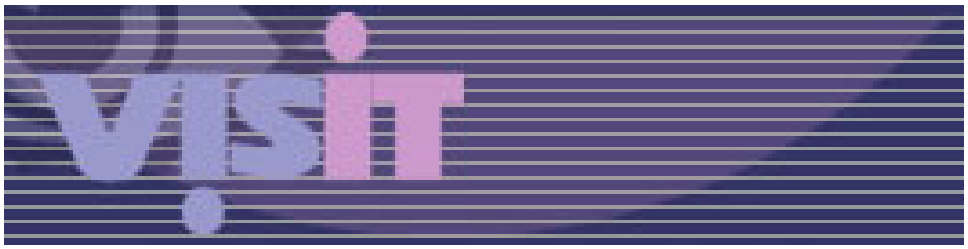
Challenge: Rapidly and smoothly update the mesh to conform to trial geometries

Impact: Used the deforming mesh metric to prototype geometry & mesh update model for potential use in SLAC accelerator design studies.

SciDAC
Scientific Discovery through Advanced Computing

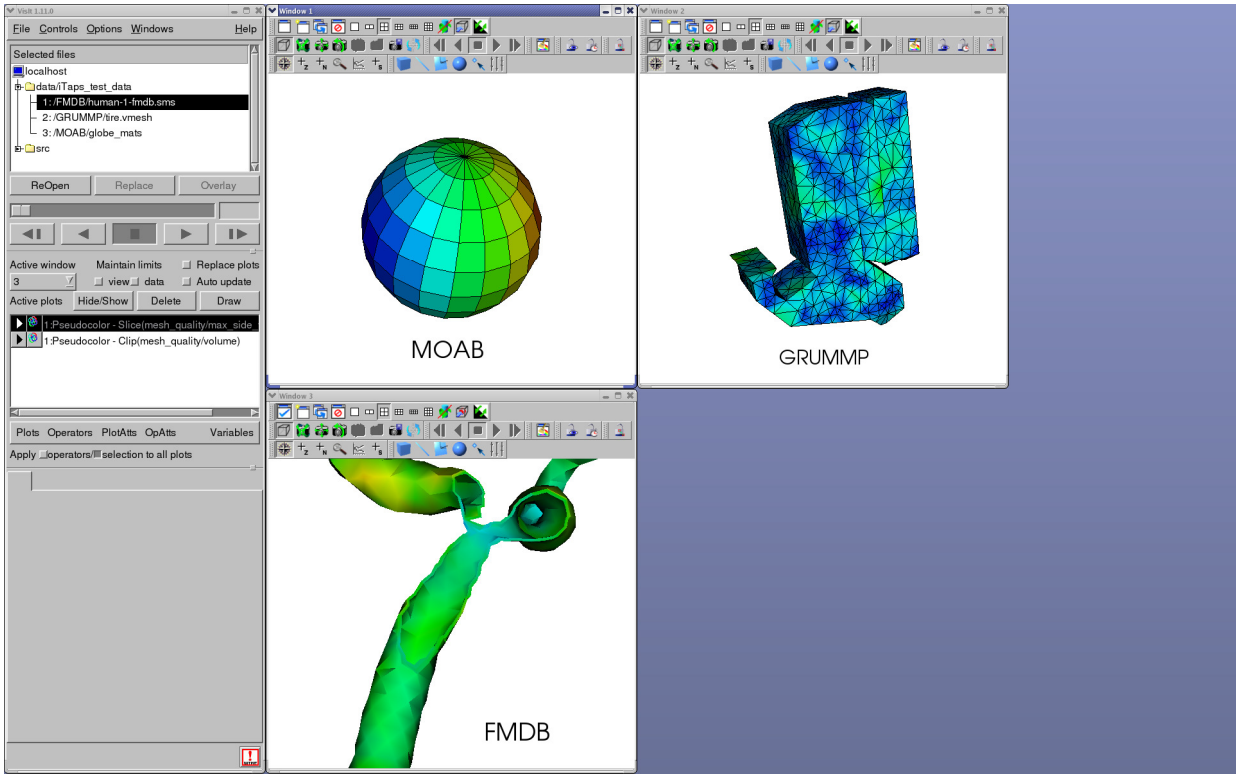
ITAPS has been integrated with VisIt as a database plug-in

- A single plug-in supports multiple ITAPS implementations
- Supports all entity types
- Supports subset and and tag data visualization
- Future integration will use VisIt's in-situ 'simulation' interface
  - Will enable any ITAPS-compliant software to integrate with VisIt at run-time



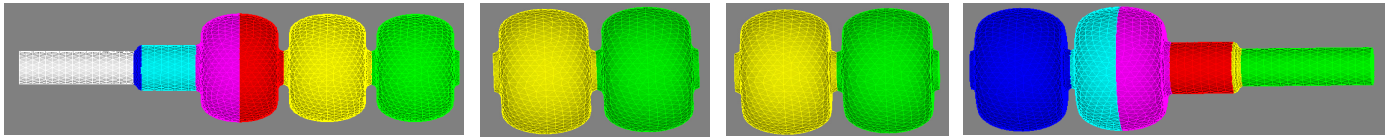
VisIt can be used to display test data from multiple implementations

Example of VisIt displaying test data from multiple ITAPS implementations simultaneously

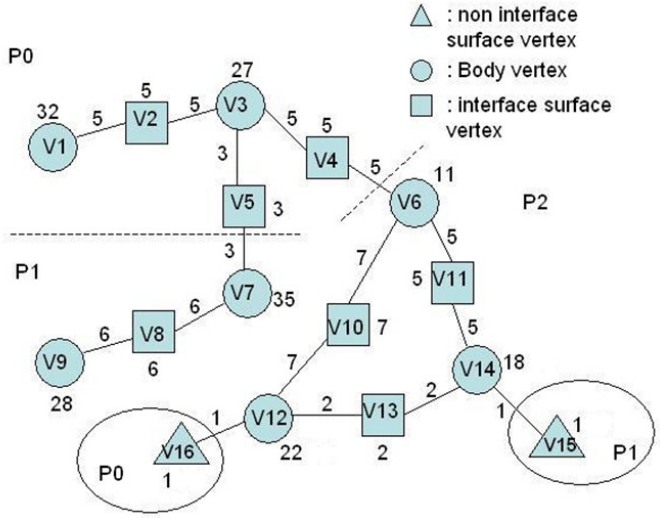


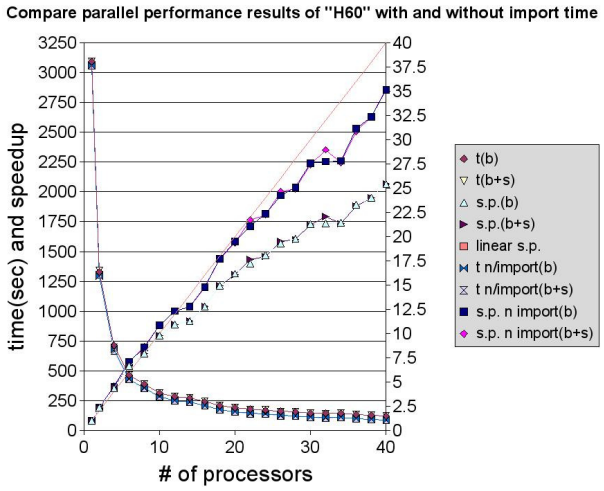
Parallel Tetrahedral Mesh Generation

- Mesh volumes *and surfaces* in parallel
- Use graph partitioning to partition work
- Use of component-based mesh, geometry, partitioning
- 1st-generation tool developed before ITAPS parallel interface specification
  - Parallel communication handled at application-level
  - Need production-capable tool



Processor: *1* *2* *3* *4*

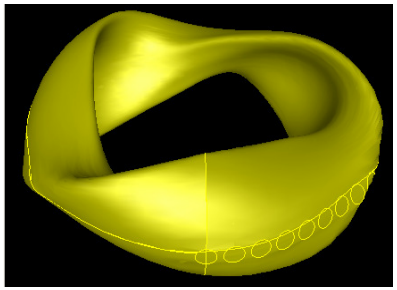




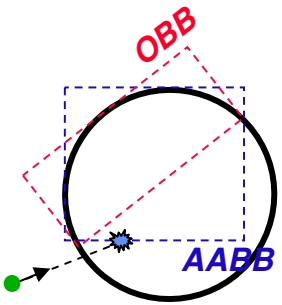
SciDAC
Scientific Discovery through Advanced Computing

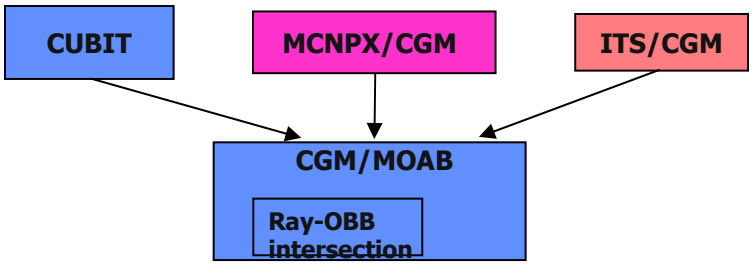
CAD-Based Monte Carlo Radiation Transport Facet-based Ray Tracing

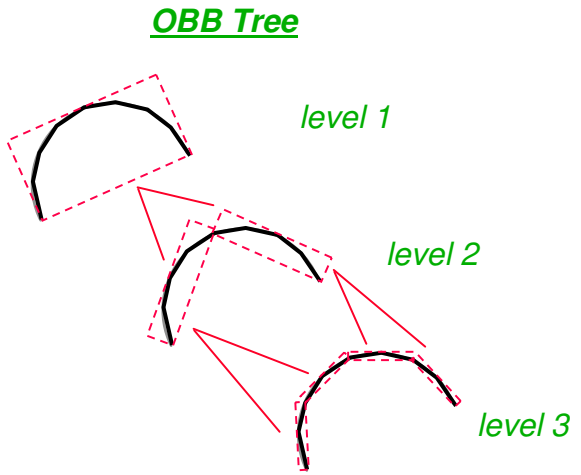
- Most Monte Carlo codes use CSG-type geometry construction
- Replaced with high-fidelity CAD modeling
  - *Use modern geom construction tools*
  - *Model more complex solids*



ARIES Compact Stellerator

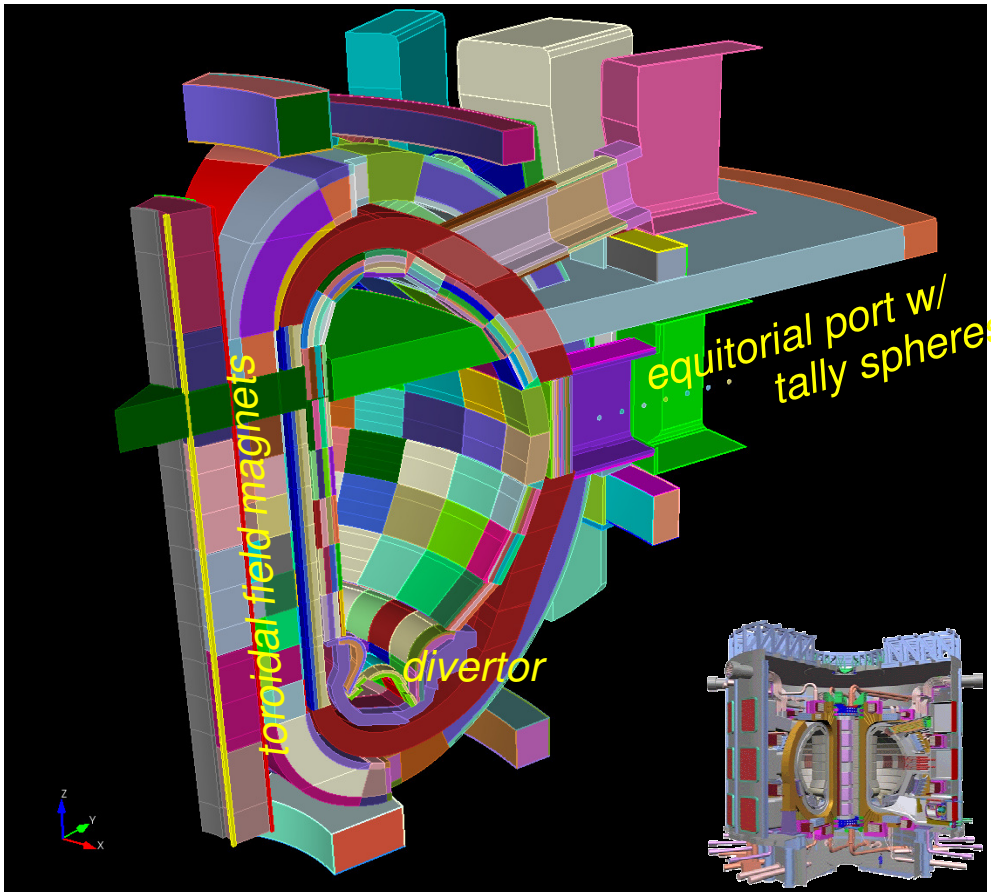




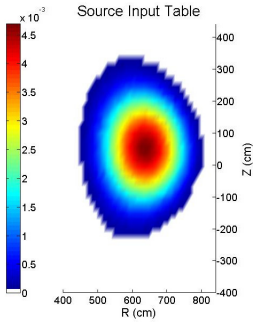


- Previous CAD-based MC 20-100x slower than native
  - Facet-based Oriented Bounding Box (OBB) tree acceleration reduced slowdown to 2-5x
- *Component-based approach simplifies introduction into multiple MC codes*
  - Incorporated into ITS in < 1 week

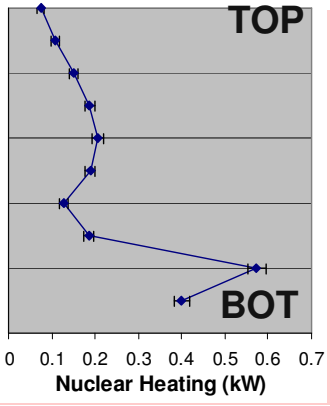
CAD-Based Monte Carlo Radiation Transport ITER Modeling & Code Comparison Effort



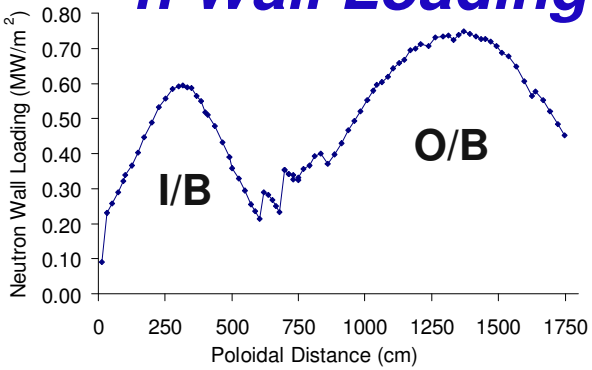
n Source



TF Magnet Heating



n Wall Loading



Joint US/FRG/PRC effort

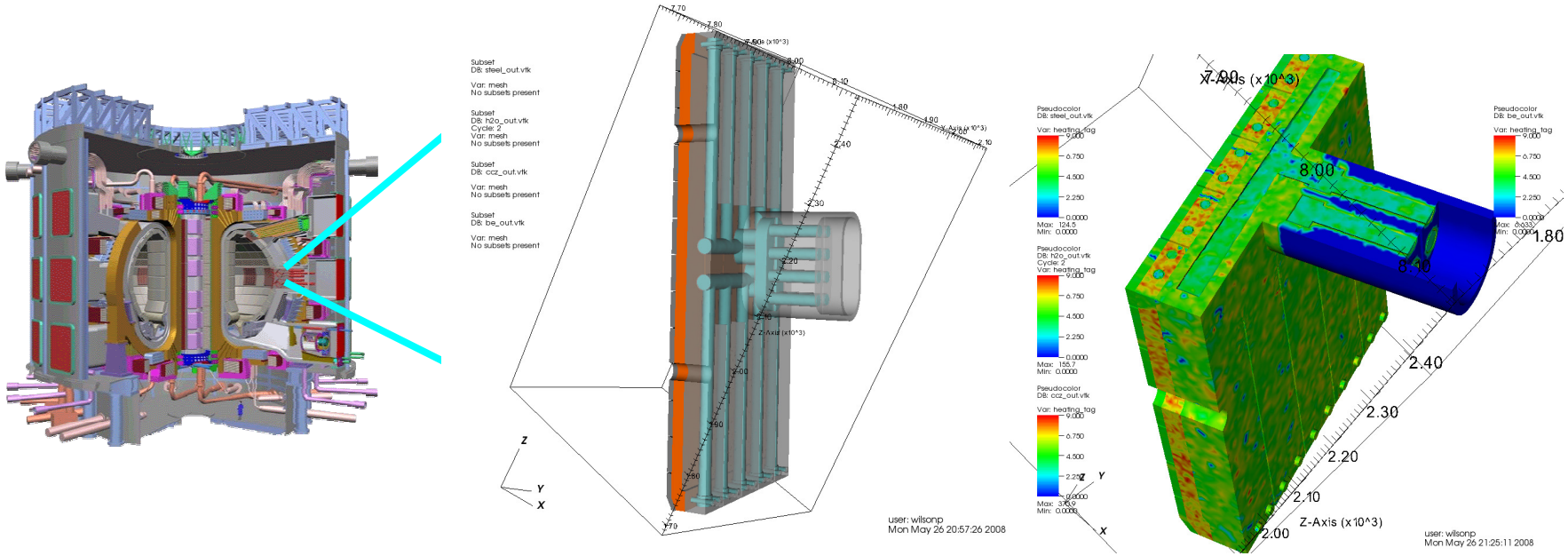
SciDAC Scientific Discovery through Advanced Computing

Sandia National Laboratories

Forschungszentrum Karlsruhe in der Helmholtz-Gemeinschaft

ASIPP

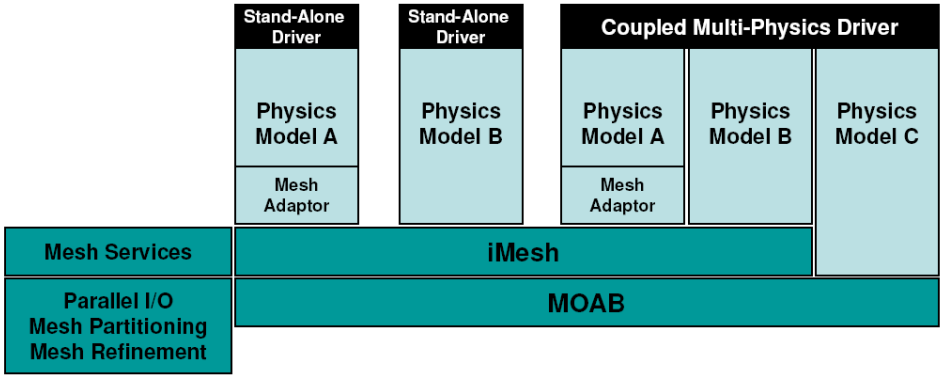
MC-Based Neutron Transport for ITER Module 13 First Wall/Shield Design



CGM→MOAB → VTK → Visit

- Heat deposition & cooling
- Helium generation close to weldable components

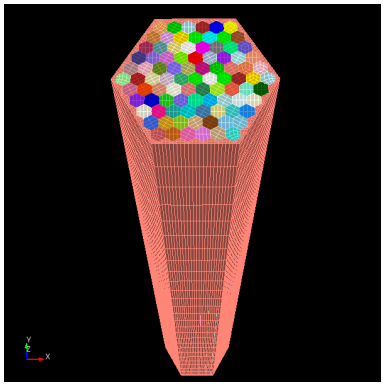
Solution Coupling

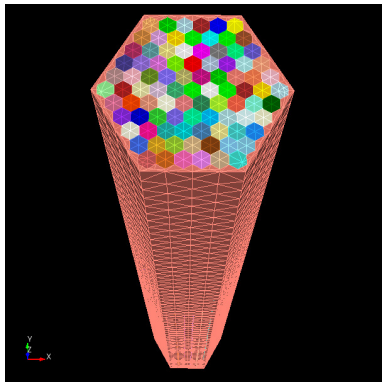


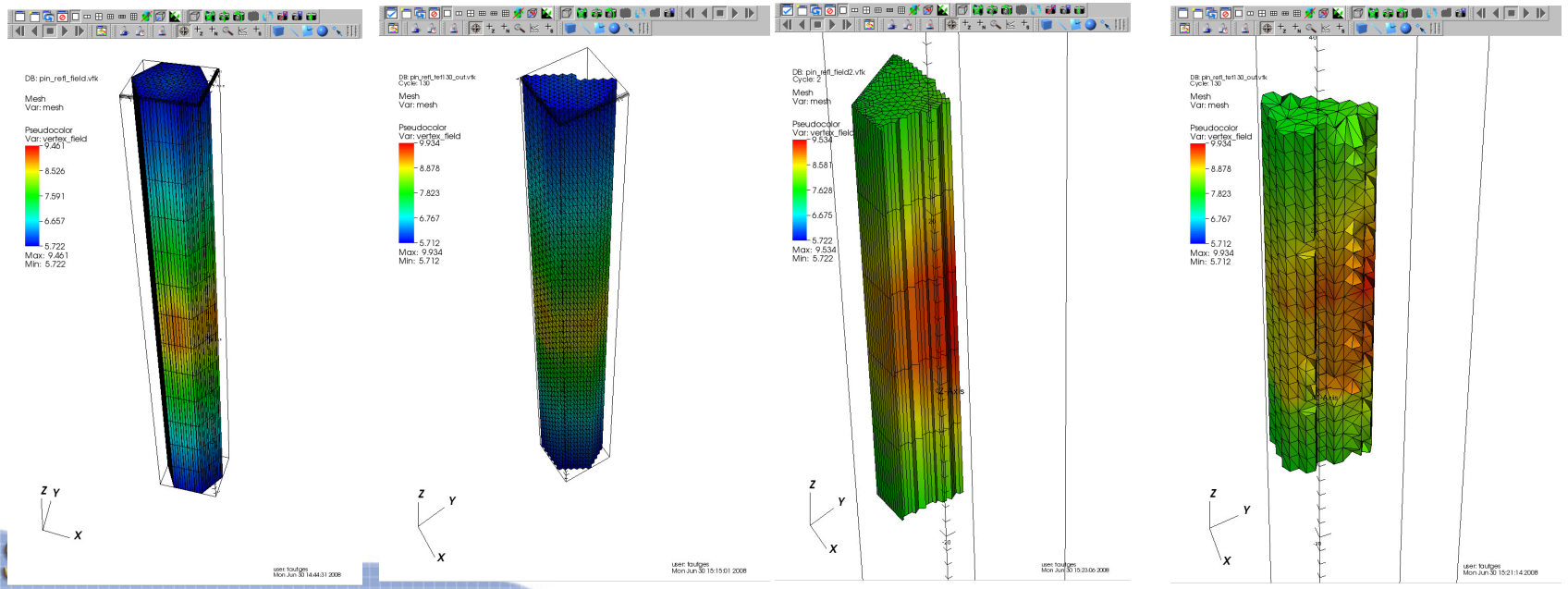
- Need to couple physics (TH, neutronics, structural mechanics) for reactor simulation
  - Need to preserve standalone development capability
- Assumptions:
  - Each physics solved on its own mesh, optimized for that physics
  - Each physics mesh distributed with its own distinct MPI communicator, independent of other mesh
  - On each processor, meshes for both/all physics stored in the same MOAB instance
- Coupling algorithm:
  - Initialization (read mesh, initialize searching structures (kdtree))
  - Point location
  - Interpolation
  - Normalization

Solution Coupling Results

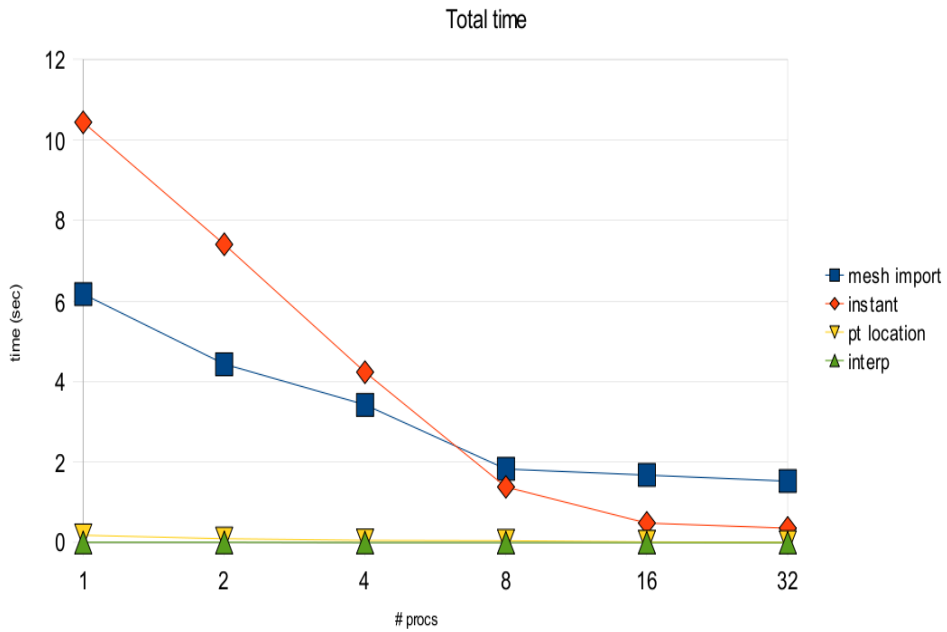
- Implemented as tool in MOAB
- 12k hex, 130k tet
- Contrived data field
- *Not* implemented on iMesh
  - *Tree-based search*
  - *Field interpolation*

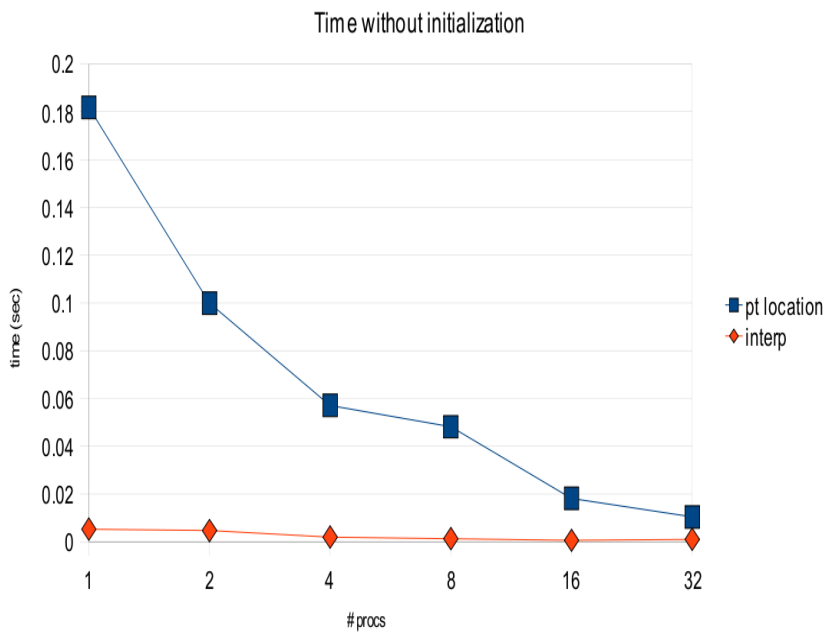






Solution Coupling Timing, Future Work





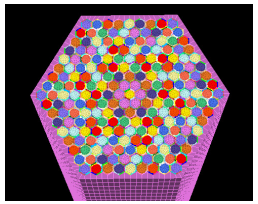
- Future work:
  - High-order (spectral) element interpolation
  - Subset-based normalization
  - Error norms
  - Timing, scalability

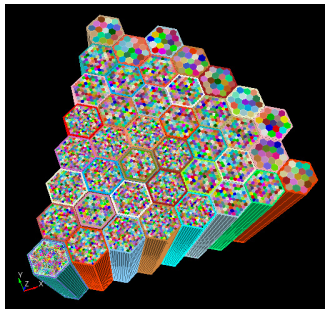
CGM Port to Open.Cascade

- CGM: provides common interface to geometry from multiple engines
  - ACIS
  - Catia/ProE (Sandia or weapons complex-restricted)
  - Facet-based geometry
- Smooths topological model variations
- Porting to open-source Open.Cascade modeler
  - Substantial improvements to OCC build process (autoconf-based)
  - Ready for (very) friendly external users (http://trac.mcs.anl.gov/projects/ITAPS/wiki/CGM)
- Methods for getting OCC-based model
  - Read OCC BREP format
  - Read/translate IGES/Step format
  - Construct model using CSG approach
    - Implemented reader for MCNP input file syntax
    - Simple thing to do same for Superfish?

ABTR 1/6 Core Mesh Generation

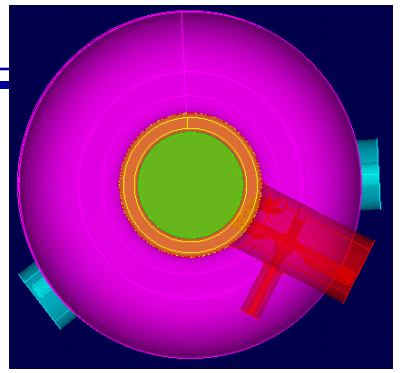
- CUBIT-based approach required 6 GB for ~5M elements
- Different ways to generate mesh
  - Start with ass'y geometry + mesh, copy/move/merge in CUBIT



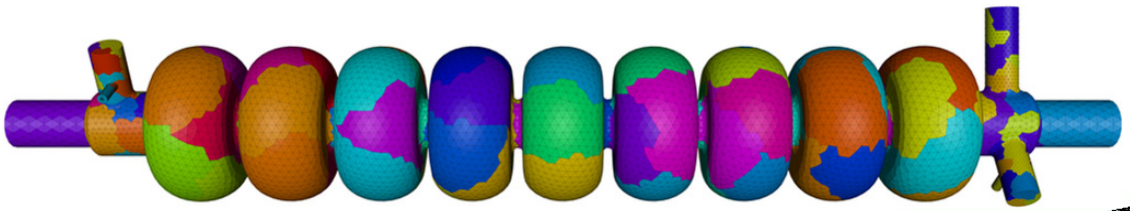


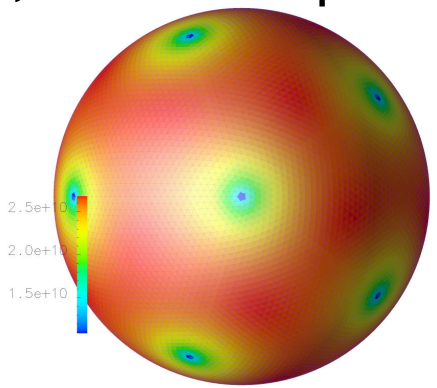
  - Start with ass'y mesh, copy/move/merge in external tool
    - Development underway in MeshKit
    - Need to handle sets carefully (material, geometric topology, BC's)
      - Copy/merge sets
  - Start with cross section 2d mesh, extrude into 3d
    - Will require extrude tool in MeshKit
    - Need to handle sets carefully: copy/expand, but with extrusion

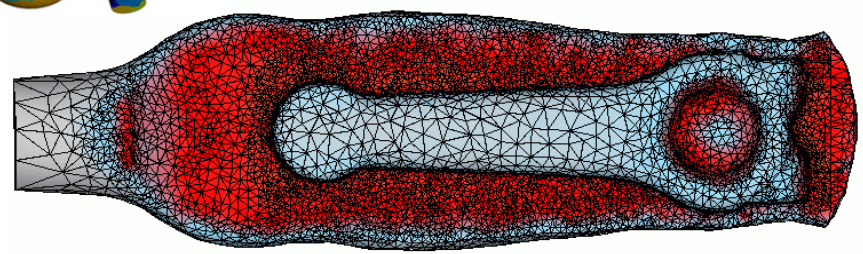
The ITAPS team has developed tools to address these needs



- CAD interaction: CGM
- Mesh generation: GRUMMP, NWGrid
- Mesh databases: FMDB, MOAB
- Mesh improvement: Mesquite, swapping tools
- Parallel Adaptive loops: FMDB, NWGrid, MeshAdapt
- Front tracking: Frontier
- Partitioning: Zoltan







SciDAC
Scientific Discovery through Advanced Computing

Opportunities for Collaboration

- ANL/ITAPS received SAP funding for a post-doc starting in November, should have it filled by end of January
- Near-term opportunities
  - Incorporation of CAD into production shape optimization, curved boundary correction tools
  - Parallel meshing
  - Fast ray-tracing, EB meshing on complex CAD
- Longer-term opportunities
  - Common set of CAD-based cavity models
  - Direct comparison of results from different codes on common mesh-based representation
  - Data analysis & viz through common mesh interface
  - ?